



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/523,877	03/13/2000	Peter Warnes	ARC.005A	6131
27299	7590	09/22/2005	EXAMINER	
GAZDZINSKI & ASSOCIATES 11440 WEST BERNARDO COURT, SUITE 375 SAN DIEGO, CA 92127			HUISMAN, DAVID J	
			ART UNIT	PAPER NUMBER
			2183	

DATE MAILED: 09/22/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary

Application No.

09/523,877

Applicant(s)

WARNES ET AL.

Examiner

David J. Huisman

Art Unit

2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --
Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 08 July 2005.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 37-40, 42, 43 and 53-83 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 37-40, 42, 43 and 53-83 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 13 March 2000 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
 - ☐ Certified copies of the priority documents have been received in Application No. _____.
 - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413) |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | Paper No(s)/Mail Date. _____ |
| 3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| Paper No(s)/Mail Date <u>04 April 2005</u> | 6) <input type="checkbox"/> Other: _____ |

10

DETAILED ACTION

1. Claims 37-40, 42-43, and 53-83 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: IDS as received on 4/4/2005 and Amendment as received on 7/8/2005.

Drawings

3. The drawings are objected to under 37 CFR 1.83(a). The drawings must show every feature of the invention specified in the claims. Therefore, the features of claims 69-71 must be shown or the feature(s) canceled from the claim(s). No new matter should be entered.

Corrected drawing sheets in compliance with 37 CFR 1.121(d) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The figure or figure number of an amended drawing should not be labeled as "amended." If a drawing figure is to be canceled, the appropriate figure must be removed from the replacement sheet, and where necessary, the remaining figures must be renumbered and appropriate changes made to the brief description of the several views of the drawings for consistency. Additional replacement sheets may be necessary to show the renumbering of the remaining figures. Each drawing sheet submitted after the filing date of an application must be labeled in the top margin as either "Replacement Sheet" or "New Sheet" pursuant to 37 CFR 1.121(d). If the changes are not accepted by the examiner, the applicant will

Art Unit: 2183

be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Claim Objections

4. Claims 68, 72, 76, and 80 are objected to because of the following informalities: In the last line of each claim, insert --at-- before “least”. Appropriate correction is required.

Maintained Rejections

5. Applicant has failed to overcome the prior art rejections set forth in the previous Office Action. Consequently, these rejections are respectfully maintained by the examiner and are copied below for applicant's convenience.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

7. Claims 37-40, 42-43, 53-59, 64-69, 71-73, 75-77, 79-81, and 83 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lee, as applied above, in view of Wirthlin et al., The Nano Processor: a Low Resource Reconfigurable Processor, 1994 (herein referred to as Wirthlin). In addition, Heuring and Jordan, “Computer Systems Design and Architecture,” 1997 (herein referred to as Heuring) is cited as extrinsic evidence for providing a definition of RISC.

Art Unit: 2183

8. Referring to claim 37, Lee has taught a pipelined digital processor comprising:

a) a branch instruction having a plurality of user-configurable modes determined by a plurality of bits controlling the execution of at least one instruction in a delay slot following the branch instruction within the pipeline, each of said modes being constrained to only one of a plurality of unique combinations of said plurality of bits. See Fig.3 and note the configurable modes, which are specified by the nullify and sign/displacement bits (Fig.2, field 507). By setting or resetting these bits, the user will configure the system to either never nullify a delay slot instruction or sometimes nullify a delay slot instruction. One basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

- 00 - conditional branch forward with delay slot execution (FDS)
- 01 - conditional branch backwards with delay slot execution (BDS)
- 10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
- 11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a different mode.

Therefore, each mode is constrained to only one unique combination of said plurality of bits.

b) Lee has not explicitly taught an extended and user-customized RISC processor having an instruction set that comprises a base instruction set and at least one extension instruction.

However, Wirthlin has taught a processor in which a base instruction set is used and an extended, custom instruction set (extension instruction) is used. See page 25, sections 3.1.1 and 3.1.2.

Furthermore, it should be realized that Wirthlin has taught a RISC processor because as defined by Heuring on page 93 (see attached), RISC (reduced instruction set computer) machines focus on reducing the number of instructions in the machine. That is, a reduced instruction set exists. Clearly, this is the case in Wirthlin because only 6 standard instructions initially exist, and these

Art Unit: 2183

instructions are hardly complex as they include load, store, add, subtract, and jump, which as is known in the art, are very common, standard instructions. See section 3.3. In addition, from the same section, Wirthlin has disclosed that the instructions are of a fixed length, which is another feature of a RISC machine (see page 93 of Heuring). Consequently, it can be seen that Wirthlin has taught an extended and user-configurable RISC processor. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a base instruction set and at least one extension instruction in a RISC processor. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task. One would have been motivated to make such a combination because Lee could very well be a RISC processor, as it is not limited to a certain type of processor, and Lee has disclosed that instructions are single-cycle instructions (Fig.5 of Lee). And, this feature is a feature of a RISC machine (see page 93 of Heuring). Consequently, since Lee may be a RISC processor (less complex according to Heuring), and Wirthlin has taught that extending a RISC processor is useful, such a combination would have been obvious.

9. Referring to claim 38, Lee has taught a pipelined digital processor comprising:
 - a) a branch instruction including two data bits defining four discrete modes controlling the execution of at least one instruction in a delay slot following the branch instruction within the

Art Unit: 2183

pipeline. See the nullify and sign/displacement bits from Fig. 3. These two bits define four delay slot modes as shown by the following breakdown (modes are in the form (nullify/sign)):

- 00 - conditional branch forward with delay slot execution (FDS)
- 01 - conditional branch backwards with delay slot execution (BDS)
- 10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
- 11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a mode with unique functionality with respect to the other three modes. Finally, see column 3, lines 58-61. By setting or clearing this nullify bit, a subsequent instruction's execution is controlled.

b) Lee has not explicitly taught an extended and user-customized RISC processor core having an instruction set that comprises a base instruction set and at least one extension instruction.

However, Wirthlin has taught a processor in which a base instruction set is used and an extended, custom instruction set (extension instruction) is used. See page 25, sections 3.1.1 and 3.1.2.

Furthermore, it should be realized that Wirthlin has taught a RISC processor because as defined by Heuring on page 93 (see attached), RISC (reduced instruction set computer) machines focus on reducing the number of instructions in the machine. That is, a reduced instruction set exists. Clearly, this is the case in Wirthlin because only 6 standard instructions initially exist, and these instructions are hardly complex as they include load, store, add, subtract, and jump, which as is known in the art, are very common, standard instructions. See section 3.3. In addition, from the same section, Wirthlin has disclosed that the instructions are of a fixed length, which is another feature of a RISC machine (see page 93 of Heuring). Consequently, it can be seen that Wirthlin has taught an extended and user-configurable RISC processor. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions

Art Unit: 2183

allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a base instruction set and at least one extension instruction in a RISC processor. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task. One would have been motivated to make such a combination because Lee could very well be a RISC processor, as it is not limited to a certain type of processor, and Lee has disclosed that instructions are single-cycle instructions (Fig.5 of Lee). And, this feature is a feature of a RISC machine (see page 93 of Heuring). Consequently, since Lee may be a RISC processor (less complex according to Heuring), and Wirthlin has taught that extending a RISC processor is useful, such a combination would have been obvious.

c) said execution is controlled without regard to a branch direction metric. The examiner agrees that when the nullify bit is on, the branch direction is taken into consideration. However, when the nullify bit is off, execution is controlled without regard to the branch direction. Therefore, the claim is still anticipated.

10. Referring to claim 39, Lee has taught a pipelined digital processor comprising:

a) a branch instruction having at least one mode controlling the execution of at least one instruction in a delay slot following the branch instruction within the pipeline using a plurality of data bits, at least one particular combination of data bits and a logical function associated therewith being adapted for assignment by a user. As shown in Fig.3, there are a plurality of modes in which each branch can operate. The mode is dependent on the nullify bit and the

Art Unit: 2183

displacement bit. Also, it should be realized that the user that is writing the program will be setting the nullify bit as well as the displacement bit by choosing whether the branch will be a forward or backwards branch. Therefore, they, along with the logical functions associated with the mode bits, are adapted for assignment by a user. Furthermore, this limitation can be interpreted as a user being able to assign a particular combination to a branch instruction, which indeed it taught by Lee, as each branch instruction will have a combination assigned to it.

b) Lee has not explicitly taught an extended and user-configured RISC processor having a user-customized processor core configuration rendered in a hardware description language model and including a base instruction set and has not taught at least one extension instruction. However, Wirthlin has taught a user-customized processor core in which a base instruction set is used and an extended, custom instruction set (extension instruction) is used. See page 25, sections 3.1.1 and 3.1.2. In addition, the core may be rendered in a hardware description language (HDL). See page 23, column 2, first full paragraph. Furthermore, it should be realized that Wirthlin has taught a RISC processor because as defined by Heuring on page 93 (see attached), RISC (reduced instruction set computer) machines focus on reducing the number of instructions in the machine. That is, a reduced instruction set exists. Clearly, this is the case in Wirthlin because only 6 standard instructions initially exist, and these instructions are hardly complex as they include load, store, add, subtract, and jump, which as is known in the art, are very common, standard instructions. See section 3.3. In addition, from the same section, Wirthlin has disclosed that the instructions are of a fixed length, which is another feature of a RISC machine (see page 93 of Heuring). Consequently, it can be seen that Wirthlin has taught an extended and user-configurable RISC processor. As disclosed by Wirthlin, the base instruction set comprises only

Art Unit: 2183

the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a base instruction set and at least one extension instruction in a RISC processor. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task. One would have been motivated to make such a combination because Lee could very well be a RISC processor, as it is not limited to a certain type of processor, and Lee has disclosed that instructions are single-cycle instructions (Fig.5 of Lee). And, this feature is a feature of a RISC machine (see page 93 of Heuring). Consequently, since Lee may be a RISC processor (less complex according to Heuring), and Wirthlin has taught that extending a RISC processor is useful, such a combination would have been obvious.

11. Referring to claim 40, Lee has taught a pipelined digital processor design comprising:

a) a branch instruction having four discrete modes controlling the execution of at least one instruction in a delay slot following the branch instruction within the pipeline, each of said modes provides unique functionality with respect to the other three modes. See the nullify and displacement bits from Fig.3. These two bits define four delay slot modes as shown by the following breakdown (modes are in the form (nullify/sign)):

- 00 - conditional branch forward with delay slot execution (FDS)
- 01 - conditional branch backwards with delay slot execution (BDS)
- 10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
- 11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a mode with unique functionality with respect to the other three modes. Finally, see column 3, lines 58-61. By setting or clearing this nullify bit, a subsequent instruction's execution is controlled.

b) Lee has not explicitly taught an extended and user-configured RISC processor having a user-customized processor core configuration rendered in a hardware description language model and an instruction set that comprises a base instruction set and at least one extension instruction.

However, Wirthlin has taught a processor in which a base instruction set is used and an extended, custom instruction set (extension instruction) is used. See page 25, sections 3.1.1 and 3.1.2. In addition, the core may be rendered in a hardware description language (HDL). See page 23, column 2, first full paragraph. Furthermore, it should be realized that Wirthlin has taught a RISC processor because as defined by Heuring on page 93 (see attached), RISC (reduced instruction set computer) machines focus on reducing the number of instructions in the machine. That is, a reduced instruction set exists. Clearly, this is the case in Wirthlin because only 6 standard instructions initially exist, and these instructions are hardly complex as they include load, store, add, subtract, and jump, which as is known in the art, are very common, standard instructions. See section 3.3. In addition, from the same section, Wirthlin has disclosed that the instructions are of a fixed length, which is another feature of a RISC machine (see page 93 of Heuring). Consequently, it can be seen that Wirthlin has taught an extended and user-configurable RISC processor. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious

Art Unit: 2183

to one of ordinary skill in the art at the time of the invention to implement a base instruction set and at least one extension instruction in a RISC processor. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task. One would have been motivated to make such a combination because Lee could very well be a RISC processor, as it is not limited to a certain type of processor, and Lee has disclosed that instructions are single-cycle instructions (Fig.5 of Lee). And, this feature is a feature of a RISC machine (see page 93 of Heuring). Consequently, since Lee may be a RISC processor (less complex according to Heuring), and Wirthlin has taught that extending a RISC processor is useful, such a combination would have been obvious.

12. Referring to claim 42, Lee in view of Wirthlin has taught a digital processor as described in claim 40. Lee has further taught that first and second of said four modes implement one- and two-cycle stalls within said pipeline, respectively. See column 5, lines 32-37, and Fig.2. Note that if a jump occurs and the displacement is positive (as shown in Fig.2, component 112), the delay slot instruction would be fetched but not executed. Therefore, in order to kill the unwanted instruction, the pipeline would be stalled for at least a single cycle. Also, see Fig.5, column 5, lines 58-63, and column 7, lines 21-25. Note that pipeline includes four stages: an address generation stage (A), a fetch stage (F), an execution stage (E), and a write stage (W). It has been disclosed that the target address of the branch is not determined until the end of the execution stage. By this time, the delay slot instruction will have been fetched, and the predicted target address will be supplied to the program counter for fetching in the next cycle. See column 7, lines 6-10. If the delay slot instruction ends up being nullified (based on the displacement and

Art Unit: 2183

nullify bits) and the predicted target is incorrect, then both instructions will need to be cancelled, resulting in a 2-cycle stall.

13. Referring to claim 43, Lee in view of Wirthlin has taught a digital processor as described in claim 42. Lee has further taught that at least one of said four modes operates without respect to a branch displacement metric. See Fig.3 and note that when the nullify bit is off, the displacement value has no part in determining that the delay slot instruction is executed.

14. Referring to claim 53, Lee in view of Wirthlin has taught a processor as described in claim 37. Lee has further taught that:

a) said bits are encoded in said branch instruction and said bits of said branch instruction comprise two data bits defining four discrete modes controlling said execution. See Fig.3 and note the configurable modes, which are specified by the nullify and sign/displacement bits (Fig.2, field 507). By setting or resetting these bits, the user will configure the system to either never nullify a delay slot instruction or sometimes nullify a delay slot instruction. One basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

- 00 - conditional branch forward with delay slot execution (FDS)
- 01 - conditional branch backwards with delay slot execution (BDS)
- 10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
- 11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a different mode.

b) said execution further being controlled without regard to a branch direction metric. When the nullify bit is off, execution is controlled without regard to the branch direction.

15. Referring to claim 54, Lee in view of Wirthlin has taught a processor as described in claim 37. Lee has further taught that said bits are encoded in said branch instruction and at least

Art Unit: 2183

one of said unique combinations of said bits of said branch instruction and a logical function associated therewith are adapted for assignment by a user. It should be noted that at the very least, the user will be specifying whether the branch is a forward or backwards branch, and therefore, the user is also specifying the displacement bit of the mode control bits. By doing this, the user is also specifying what functionality is to occur with such a displacement bit.

Furthermore, this limitation can be interpreted as a user being able to assign a particular combination to a branch instruction, which indeed is taught by Lee, as each branch instruction will have a combination assigned to it.

16. Referring to claim 55, Lee in view of Wirthlin has taught a processor as described in claim 53. Lee has further taught that at least one of said unique combinations of said bits of said branch instruction and a logical function associated therewith are adapted for assignment by a user. It should be noted that at the very least, the user will be specifying whether the branch is a forward or backwards branch, and therefore, the user is also specifying the displacement bit of the mode control bits. By doing this, the user is also specifying what functionality is to occur with such a displacement bit. Furthermore, this limitation can be interpreted as a user being able to assign a particular combination to a branch instruction, which indeed is taught by Lee, as each branch instruction will have a combination assigned to it.

17. Referring to claim 56, Lee in view of Wirthlin has taught a processor as described in claim 38. Lee has further taught that each of said four discrete modes is constrained to only one of a plurality of unique combinations of said two bits. One basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)

Art Unit: 2183

- 01 - conditional branch backwards with delay slot execution (BDS)
- 10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
- 11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized from this breakdown that each combination of bits corresponds to a different mode. Therefore, each mode is constrained to only one unique combination of said plurality of bits.

18. Referring to claim 57, Lee in view of Wirthlin has taught a processor as described in claim 38. Lee has further taught that said bits are encoded in said branch instruction and at least one of said unique combinations of said bits of said branch instruction and a logical function associated therewith are adapted for assignment by a user. It should be noted that at the very least, the user will be specifying whether the branch is a forward or backwards branch, and therefore, the user is also specifying the displacement bit of the mode control bits. By doing this, the user is also specifying what functionality is to occur with such a displacement bit.

Furthermore, this limitation can be interpreted as a user being able to assign a particular combination to a branch instruction, which indeed is taught by Lee, as each branch instruction will have a combination assigned to it.

19. Referring to claim 58, Lee in view of Wirthlin has taught a processor as described in claim 56. Lee has further taught that said bits are encoded in said branch instruction and at least one of said combinations of said bits of said branch instruction and a logical function associated therewith are adapted for assignment by a user. It should be noted that at the very least, the user will be specifying whether the branch is a forward or backwards branch, and therefore, the user is also specifying the displacement bit of the mode control bits. By doing this, the user is also specifying what functionality is to occur with such a displacement bit. Furthermore, this

Art Unit: 2183

limitation can be interpreted as a user being able to assign a particular combination to a branch instruction, which indeed is taught by Lee, as each branch instruction will have a combination assigned to it.

20. Referring to claim 59, Lee has taught a processor having at least one pipeline comprising at least instruction fetch, decode, and execute stages (see Fig.5 and column 6, lines 56-63, and note the explicit disclosure of fetch and execute stages. Although a decode stage is not explicitly mentioned, decoding of instructions is a step that must be performed. Decoding is disclosed in column 7, lines 16-20, and column 1, lines 23-26. Decoding will occur at some point in the pipeline and that will be the decode stage) and an associated data storage device, wherein the execution of instructions within said at least one pipeline is controlled by the method comprising:

a) storing an instruction set within said data storage device, said instruction set comprising a plurality of instruction words, each of said instruction words comprising a plurality of data bits, at least one of said instruction words comprising a user-configurable branch instruction having a plurality of unique functional modes exclusively associated with respective ones of unique combinations of a plurality of mode control bits, said branch instruction directing branching to a first address within said data storage device. Lee has taught an instruction set with a plurality of instructions that would inherently comprise a plurality of bits and would inherently be stored in a data storage device in order to processor-accessible and executable. Furthermore, it is inherent that a branch will cause a jump to a specified address within the data storage device. Finally, it should be noted that these branch instructions are user-configurable instructions having unique functional modes associated with unique mode control bit combinations. For instance, one basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

Art Unit: 2183

- 00 - conditional branch forward with delay slot execution (FDS)
- 01 - conditional branch backwards with delay slot execution (BDS)
- 10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
- 11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a unique delay slot mode; that is, different functionality is achieved with each combination of bits.

b) assigning one of a plurality of values to each of said mode control bits of said at least one branch instruction. See column 3, lines 46-51. Lee discloses that each instruction contains 32 data bits of information. This information includes source registers, displacements, an opcode, condition fields, and a nullify bit and a displacement sign bit (mode control bits). Each of these bits is assigned a value (0 or 1). For instance, if the user decides to use a branch instruction, then he/she must set the nullify bit to the desired state and the displacement bit will be determined based on whether the user specifies the branch to be a forward or backwards branch.

c) decoding said at least one branch instruction including said assigned values. Since, the values of the nullify bit and displacement sign bit are part of each branch instruction, it follows that the value will be decoded as the branch instruction is decoded. See Fig.1 and column 3, lines 46-51.

d) determining whether to execute an instruction within said pipeline in a stage preceding that of said at least one branch instruction based at least in part on said assigned values. The delay slot instruction would be in a pipeline stage preceding that of the branch instruction. And, Lee discloses a system in which the execution of the delay slot instruction is determined based on the value of the nullify bit and/or displacement bit.

Art Unit: 2183

e) branching to said first address based on said at least one branching instruction. Recall from claim 16, that it is the inherent nature of a branch instruction (when taken) to jump to a specified address.

f) performing, based at least in part on said act of decoding said assigned values, at least one other function dictated by the unique functional mode associated with said assigned values.

Looking at Fig.2, component 113, and Fig.3, it should be realized that depending on the mode of the branch instruction, the delay slot instruction will either be executed or not executed. If it is determined that the delay slot instruction will be executed, then it will eventually be performed following the branch. If the delay slot instruction is an "ADD" instruction, for instance, an addition function will be performed.

g) Lee has not explicitly taught a user-customized and user-extended RISC processor core wherein said processor core has a configuration determined at least in part by said user-customization and user-extension at the time of its design. However, Wirthlin has taught a processor in which a base instruction set is used and an extended, custom instruction set (extension instruction) is used. See page 25, sections 3.1.1 and 3.1.2. In addition, the core may be rendered in a hardware description language (HDL). See page 23, column 2, first full paragraph. Furthermore, it should be realized that Wirthlin has taught a RISC processor because as defined by Heuring on page 93 (see attached), RISC (reduced instruction set computer) machines focus on reducing the number of instructions in the machine. That is, a reduced instruction set exists. Clearly, this is the case in Wirthlin because only 6 standard instructions initially exist, and these instructions are hardly complex as they include load, store, add, subtract, and jump, which as is known in the art, are very common, standard instructions. See section 3.3.

Art Unit: 2183

In addition, from the same section, Wirthlin has disclosed that the instructions are of a fixed length, which is another feature of a RISC machine (see page 93 of Heuring). Consequently, it can be seen that Wirthlin has taught an extended and user-configurable RISC processor. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a base instruction set and at least one extension instruction in a RISC processor. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task. One would have been motivated to make such a combination because Lee could very well be a RISC processor, as it is not limited to a certain type of processor, and Lee has disclosed that instructions are single-cycle instructions (Fig.5 of Lee). And, this feature is a feature of a RISC machine (see page 93 of Heuring). Consequently, since Lee may be a RISC processor (less complex according to Heuring), and Wirthlin has taught that extending a RISC processor is useful, such a combination would have been obvious.

21. Referring to claims 64-67, claims 64-67 are essentially the same as claims 37-40 except that claims 64-67 also include the limitation that "said core is user-customized at time of its design". This is deemed inherent by the examiner because the design process is a customization process. Furthermore, claims 64-67 are rejected for the same reasons set forth in the rejections of claims 37-40, respectively.

Art Unit: 2183

22. Referring to claim 68, claim 68 is the same a superset of claim 37. Consequently, claim 68 is partially rejected for the same reasons set forth in the rejection of claim 37. Claim 68 further requires that said processor core is user-customized by at least:

a) receiving one or more inputs from a user for at least one customized parameter of said processor. The examiner asserts that this is inherent at least within Wirthlin which states that a core may be programmed using HDL. In HDL coding, a user will link together multiple coded components which are customized by the user. For instance, if a processor is to include an ALU, the user will code the ALU based on his/her specification (size of the ALU, inputs to the ALU, what operations will be performed by the ALU, etc.).

b) generating through an automated process a customized description language model of said core based on the least one customized parameter. The examiner asserts that this is also inherent because a model of the core will be generated automatically by an HDL compiler based on the HDL code written by the user. With this model, the user can see how the core will operate under a certain set of conditions.

23. Referring to claim 69, Lee in view of Wirthlin has taught a core as described in claim 68. Lee in view of Wirthlin has not explicitly taught that said automated process comprises modifying at least one prototype description by substituting values in the at least one prototype description or merging additional descriptions based on the at least one customized parameter. However, Official Notice is taken that multiple descriptions may be merged together to form a model of an overall core. For instance, one description may simulate a register file while another simulates an ALU. Since these two components would operate together in a working processor, they could be merged together in an HDL model based on the user's specification so that their

Art Unit: 2183

interaction may be simulated. Clearly, the more components merged in a model for simulation and testing will give a better idea of how an actual system including the merged components would operate. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Lee in view of Wirthlin such that multiple descriptions are merged in an HDL model.

24. Referring to claim 71, Lee in view of Wirthlin has taught a core as described in claim 68. It has been further taught that the customized description language model includes both functional and structural description language descriptions for the processor core. This is deemed inherent by the examiner as applicant is claiming a hardware description language (HDL). That is, an HDL will define the structure and coupling of circuits and how they function in response to specified signals.

25. Referring to claim 72, claim 72 is rejected for the same reasons set forth in the rejections of claims 38 and 68.

26. Referring to claim 73, Lee in view of Wirthlin has taught a core as described in claim 72. Furthermore, claim 73 is rejected for the same reasons set forth in the rejection of claim 69.

27. Referring to claim 75, Lee in view of Wirthlin has taught a core as described in claim 72. Furthermore, claim 75 is rejected for the same reasons set forth in the rejection of claim 71.

28. Referring to claim 76, claim 76 is rejected for the same reasons set forth in the rejections of claims 39 and 68.

29. Referring to claim 77, Lee in view of Wirthlin has taught a core as described in claim 76. Furthermore, claim 77 is rejected for the same reasons set forth in the rejection of claim 69.

Art Unit: 2183

30. Referring to claim 79, Lee in view of Wirthlin has taught a core as described in claim 76.

Furthermore, claim 79 is rejected for the same reasons set forth in the rejection of claim 71.

31. Referring to claim 80, claim 80 is rejected for the same reasons set forth in the rejections of claims 40 and 68.

32. Referring to claim 81, Lee in view of Wirthlin has taught a core as described in claim 80.

Furthermore, claim 81 is rejected for the same reasons set forth in the rejection of claim 69.

33. Referring to claim 83, Lee in view of Wirthlin has taught a core as described in claim 80.

Furthermore, claim 83 is rejected for the same reasons set forth in the rejection of claim 71.

34. Claims 60-63 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lee in view of Wirthlin in view of Heuring, as applied above, and further in view of Kawasaki et al., U.S. Patent No. 5,530,965 (herein referred to as Kawasaki).

35. Referring to claim 60, Lee has taught a method of controlling the execution of instructions within a pipelined processor, comprising:

a) providing an instruction set comprising a plurality of instruction words. Lee discloses in column 3, lines 46-47, that each instruction within the instruction set contains a 6-bit opcode, which means a total of 64 instructions could exist within the system. Lee also discloses in column 6, lines 36-39, that the system contains a floating-point unit, which means floating-point instructions would exist.

b) each of said instruction words comprising a plurality of data bits. See column 3, lines 40-42. Each instruction is 32 bits.

Art Unit: 2183

c) at least one of said words comprising a jump instruction having at least one user-configurable mode and at least one user-definable mode associated therewith, said user-configurable and user-definable modes each being specified by the same ones of said plurality of data bits, said at least one user-definable mode not being predetermined in terms of function. See Fig.2, component 102, and note the use of a branch (jump) instruction. Also, it should be realized that that the user can configure a jump instruction such that its delay slot instruction is always executed by turning the nullify bit off (see Fig.3 and Fig.1, field 507). This is equivalent to at least one user-configurable mode. In addition, the user can define the delay slot of an instruction to either execute or not execute by turning on the nullify bit (see Fig.3 and Fig.1, field 507). With the nullify bit on, the functionality is not predetermined because whether or not the delay slot instruction executes is partially dependent on whether the associated branch is taken or not taken, and a branch outcome is determined as the program is running. So the functionality will be determined while the program is running. This is equivalent to at least one user-definable mode. It can be seen that this configurability and definability are specified by the same bits (i.e., the bits specifically used by branch instructions - Fig.2, fields 507 and 508, but specifically, the nullify bit).

d) assigning one of a plurality of values to said ones of said data bits of said at least one jump instruction. See column 3, lines 46-51. Lee discloses that each branch instruction contains a nullify bit and a displacement sign bit. These bits can be assigned a value (0 or 1) as shown in Fig.3.

e) controlling the execution of at least one subsequent instruction within said pipeline based on said one assigned value of said ones of data bits when said at least one jump instruction is

Art Unit: 2183

decoded. See column 3, lines 58-61. By setting or clearing this nullify bit, a subsequent instruction's execution is controlled.

f) wherein said method further comprises generating a long immediate constant using a single word instruction by:

f1) Lee has further taught of providing an instruction word having an opcode and at least one short immediate value associated therewith, said at least one short immediate value comprising a plurality of bits. Note from column 3, lines 46-51, that the instruction format includes an 11 bit displacement field, which holds an immediate constant for branching purposes.

f2) Lee has not explicitly taught selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register. However, Kawasaki has taught such a concept for branching purposes as well as for other instructions, such as move instructions. See column 51, lines 50-55. Kawasaki has disclosed a move instruction of the format: **mov #imm, Rn**, where the short immediate value specified by #imm is sign-extended to form a long immediate value which is then stored in the register specified by Rn. By sign-extending an immediate value, a portion (the sign bit) of the value is copied into the most significant bit positions of the long immediate value. For instance, if the #imm field specified a short 4-bit immediate value 1010, which is to be transformed into an 8-bit long immediate value, then 1010 would be sign-extended to 1111010, where the sign bit of the 4-bit value is copied into the 4 most significant bit

positions of the long value. It also follows that the 4-bit value has been shifted. Note that initially, 1010 contained a 1 in the most significant bit position, a 0 in the second most significant bit position, a 1 in the third most significant bit position, and a 0 in the fourth most significant bit position. After sign-extending the 4-bit value, the same numbers become the fifth, sixth, seventh, and eighth most significant bits, respectively. Hence, they have been shifted. Furthermore, in column 42, lines 16-27, Kawasaki has disclosed that this type of move instruction is used to help branch to addresses out of the short immediate value's range. More specifically, if a branch needs to branch further than what the short displacement allows, then the destination address is moved to the register specified by the "mov" instruction and a "jmp" instruction (shown in column 48, lines 28-30) is used with to jump to the address stored in the register. A person of ordinary skill in the art would have recognized that this concept could be applicable in a system that is concerned with branching, such as Lee's. Such a concept would allow a branch instruction to branch to an address outside of the range of just a short immediate displacement value. This in turn would give a programmer more freedom in that they would not have to worry about program length or certain parts of a program being out of reach of a branch. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to take Kawasaki's concept of selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register, and apply it to the system of Lee.

Art Unit: 2183

g) Lee has not explicitly taught a user-customized and user-extended RISC processor, said processor further being generated from a hardware description language model. However, Wirthlin has taught a processor in which a base instruction set is used and an extended, custom instruction set (extension instruction) is used. See page 25, sections 3.1.1 and 3.1.2. In addition, the core may be rendered in a hardware description language (HDL). See page 23, column 2, first full paragraph. Furthermore, it should be realized that Wirthlin has taught a RISC processor because as defined by Heuring on page 93 (see attached), RISC (reduced instruction set computer) machines focus on reducing the number of instructions in the machine. That is, a reduced instruction set exists. Clearly, this is the case in Wirthlin because only 6 standard instructions initially exist, and these instructions are hardly complex as they include load, store, add, subtract, and jump, which as is known in the art, are very common, standard instructions. See section 3.3. In addition, from the same section, Wirthlin has disclosed that the instructions are of a fixed length, which is another feature of a RISC machine (see page 93 of Heuring). Consequently, it can be seen that Wirthlin has taught an extended and user-configurable RISC processor. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a base instruction set and at least one extension instruction in a RISC processor. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task. One would have been motivated to

Art Unit: 2183

make such a combination because Lee could very well be a RISC processor, as it is not limited to a certain type of processor, and Lee has disclosed that instructions are single-cycle instructions (Fig.5 of Lee). And, this feature is a feature of a RISC machine (see page 93 of Heuring). Consequently, since Lee may be a RISC processor (less complex according to Heuring), and Wirthlin has taught that extending a RISC processor is useful, such a combination would have been obvious.

36. Referring to claim 61, Lee has taught a method of controlling the execution of instructions within a pipelined processor, comprising:

- a) providing an instruction set comprising a plurality of instruction words. Lee discloses in column 3, lines 46-47, that each instruction within the instruction set contains a 6-bit opcode, which means a total of 64 instructions could exist within the system. Lee also discloses in column 6, lines 36-39, that the system contains a floating-point unit, which means floating-point instructions would exist.
- b) each of said instruction words comprising a plurality of data bits. See column 3, lines 40-42. Each instruction is 32 bits.
- c) at least one of said words comprising a jump instruction having at least one user-configurable mode and at least one user-definable mode associated therewith, said user-configurable and user-definable modes each being specified by the same ones of said plurality of data bits, said at least one user-definable mode not being predetermined in terms of function. See Fig.2, component 102, and note the use of a branch (jump) instruction. Also, it should be realized that that the user can configure a jump instruction such that its delay slot instruction is always executed by turning the nullify bit off (see Fig.3 and Fig.1, field 507). This is equivalent to at least one user-

Art Unit: 2183

configurable mode. In addition, the user can define the delay slot of an instruction to either execute or not execute by turning on the nullify bit (see Fig.3 and Fig.1, field 507). With the nullify bit on, the functionality is not predetermined because whether or not the delay slot instruction executes is partially dependent on whether the associated branch is taken or not taken, and a branch outcome is determined as the program is running. So the functionality will be determined while the program is running. This is equivalent to at least one user-definable mode. It can be seen that this configurability and definability are specified by the same bits (i.e., the bits specifically used by branch instructions - Fig.2, fields 507 and 508, but specifically, the nullify bit).

d) assigning one of a plurality of values to said ones of said data bits of said at least one jump instruction. See column 3, lines 46-51. Lee discloses that each branch instruction contains a nullify bit and a displacement sign bit. These bits can be assigned a value (0 or 1) as shown in Fig.3.

e) controlling the execution of at least one subsequent instruction within said pipeline based on said one assigned value of said ones of data bits when said at least one jump instruction is decoded. See column 3, lines 58-61. By setting or clearing this nullify bit, a subsequent instruction's execution is controlled.

f) wherein at least one of said plurality of instruction words comprises an op-code (column 3, lines 46-47), a plurality of fields (Fig.1), each of said fields comprising a plurality of bits (see column 3, lines 46-51), said at least one instruction word being encoded according to the method comprising:

Art Unit: 2183

f1) associating a first of said fields with a first data source. See column 3, lines 47-48 (component 503).

f2) associating a second of said fields with a second data source. See column 3, lines 48-49 (component 504).

f3) performing a logical operation using said first and second data sources as operands, said logical operation being specified by said op-code. See column 3, lines 51-55. In this case, the opcode specifies a compare and branch instruction where the comparison is performed between the contents of the two specified registers.

g) Lee has not taught that said method further comprises generating a long immediate constant using a single word instruction by selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register. However, Kawasaki has taught such a concept for branching purposes as well as for other instructions, such as move instructions. See column 51, lines 50-55. Kawasaki has disclosed a move instruction of the format: **mov #imm, Rn**, where the short immediate value specified by #imm is sign-extended to form a long immediate value which is then stored in the register specified by Rn. By sign-extending an immediate value, a portion (the sign bit) of the value is copied into the most significant bit positions of the long immediate value. For instance, if the #imm field specified a short 4-bit immediate value 1010, which is to be transformed into an 8-bit long immediate value, then 1010 would be sign-extended to 11111010, where the sign bit of the 4-bit value is copied into the 4 most significant bit positions of the long value. It also follows that the 4-bit value has been shifted. Note that

Art Unit: 2183

initially, 1010 contained a 1 in the most significant bit position, a 0 in the second most significant bit position, a 1 in the third most significant bit position, and a 0 in the fourth most significant bit position. After sign-extending the 4-bit value, the same numbers become the fifth, sixth, seventh, and eighth most significant bits, respectively. Hence, they have been shifted.

Furthermore, in column 42, lines 16-27, Kawasaki has disclosed that this type of move instruction is used to help branch to addresses out of the short immediate value's range. More specifically, if a branch needs to branch further than what the short displacement allows, then the destination address is moved to the register specified by the "mov" instruction and a "jmp" instruction (shown in column 48, lines 28-30) is used with to jump to the address stored in the register. A person of ordinary skill in the art would have recognized that this concept could be applicable in a system that is concerned with branching, such as Lee's. Such a concept would allow a branch instruction to branch to an address outside of the range of just a short immediate displacement value. This in turn would give a programmer more freedom in that they would not have to worry about program length or certain parts of a program being out of reach of a branch. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to take Kawasaki's concept of selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register, and apply it to the system of Lee.

h) Lee has not explicitly taught a user-customized and user-extended RISC processor, said processor further being generated from a hardware description language model. However, Wirthlin has taught a processor in which a base instruction set is used and an extended, custom

Art Unit: 2183

instruction set (extension instruction) is used. See page 25, sections 3.1.1 and 3.1.2. In addition, the core may be rendered in a hardware description language (HDL). See page 23, column 2, first full paragraph. Furthermore, it should be realized that Wirthlin has taught a RISC processor because as defined by Heuring on page 93 (see attached), RISC (reduced instruction set computer) machines focus on reducing the number of instructions in the machine. That is, a reduced instruction set exists. Clearly, this is the case in Wirthlin because only 6 standard instructions initially exist, and these instructions are hardly complex as they include load, store, add, subtract, and jump, which as is known in the art, are very common, standard instructions. See section 3.3. In addition, from the same section, Wirthlin has disclosed that the instructions are of a fixed length, which is another feature of a RISC machine (see page 93 of Heuring). Consequently, it can be seen that Wirthlin has taught an extended and user-configurable RISC processor. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a base instruction set and at least one extension instruction in a RISC processor. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task. One would have been motivated to make such a combination because Lee could very well be a RISC processor, as it is not limited to a certain type of processor, and Lee has disclosed that instructions are single-cycle instructions (Fig.5 of Lee). And, this feature is a feature of a RISC machine (see page 93 of Heuring).

Art Unit: 2183

Consequently, since Lee may be a RISC processor (less complex according to Heuring), and Wirthlin has taught that extending a RISC processor is useful, such a combination would have been obvious.

37. Referring to claim 62, Lee has taught a processor comprising:

a) a user-customized processor core having a multistage instruction pipeline, said core being adapted to decode and execute an instruction set comprising a plurality of instruction words.

Fig.5 shows a 4-stage pipeline that is further described in column 6, line 56, to column 7, line 39.

Furthermore, it is inherent that the processor will decode and execute multiple instructions (as established in the rejection of claim 1) from an instruction set. Finally, it should be realized that all processors are user-customized as processors are components designed by users.

b) a data interface between said processor core and an information storage device. It is inherent that in order for a processor to execute instructions, they must be stored in the processor's memory. Therefore, the instructions would be stored in an information storage device that is directly accessible by the processor.

c) an instruction set comprising a plurality of instruction words, at least one of said instruction words being a user-configurable jump instruction containing data defining a plurality of jump delay slot modes and at least one user-defined mode, said jump delay slot modes and at least one user-defined mode each being specified by the same portions of said data, said at least one user-defined mode not being predetermined in terms of function, said plurality of modes controlling the execution of instructions within said instruction pipeline of said processor core in response to said at least one jump instruction word within said instruction set. See Fig.2, component 102, and note the use of a branch (jump) instruction. Furthermore, Fig.3 shows that the branches are

Art Unit: 2183

user-configurable in that a number of different modes which are achieved in part by the user.

For example, the user can configure a jump instruction such that its delay slot instruction is always executed by turning the nullify bit off (see Fig.3 and Fig.1, field 507). In addition, the user can define the delay slot of an instruction to either execute or not execute by turning on the nullify bit (see Fig.3 and Fig.1, field 507). With the nullify bit on, the functionality is not predetermined because whether or not the delay slot instruction executes or not is partially dependent on whether the associated branch is taken or not taken, and a branch outcome is determined as the program is running. So the functionality will be determined while the program is running. This is equivalent to at least one user-definable mode. It can be seen that this configurability and definability are specified by the same bits (i.e., the bits specifically used by branch instructions - Fig.2, fields 507 and 508, but specifically, the nullify bit).

d) wherein said processor is further adapted to generate a long immediate constant using a single word instruction by:

f1) Lee has further taught of providing an instruction word having an opcode and at least one short immediate value associated therewith, said at least one short immediate value comprising a plurality of bits. Note from column 3, lines 46-51, that the instruction format includes an 11 bit displacement field, which holds an immediate constant for branching purposes.

f2) Lee has not explicitly taught selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register. However, Kawasaki has

Art Unit: 2183

taught such a concept for branching purposes as well as for other instructions, such as move instructions. See column 51, lines 50-55. Kawasaki has disclosed a move instruction of the format: **mov #imm, Rn**, where the short immediate value specified by #imm is sign-extended to form a long immediate value which is then stored in the register specified by Rn. By sign-extending an immediate value, a portion (the sign bit) of the value is copied into the most significant bit positions of the long immediate value. For instance, if the #imm field specified a short 4-bit immediate value 1010, which is to be transformed into an 8-bit long immediate value, then 1010 would be sign-extended to 11111010, where the sign bit of the 4-bit value is copied into the 4 most significant bit positions of the long value. It also follows that the 4-bit value has been shifted. Note that initially, 1010 contained a 1 in the most significant bit position, a 0 in the second most significant bit position, a 1 in the third most significant bit position, and a 0 in the fourth most significant bit position. After sign-extending the 4-bit value, the same numbers become the fifth, sixth, seventh, and eighth most significant bits, respectively. Hence, they have been shifted. Furthermore, in column 42, lines 16-27, Kawasaki has disclosed that this type of move instruction is used to help branch to addresses out of the short immediate value's range. More specifically, if a branch needs to branch further than what the short displacement allows, then the destination address is moved to the register specified by the "mov" instruction and a "jmp" instruction (shown in column 48, lines 28-30) is used with to jump to the address stored in the register. A person of ordinary skill in the art would have recognized that this concept could be applicable in a system that is concerned with branching, such as Lee's. Such a concept would allow a branch

Art Unit: 2183

instruction to branch to an address outside of the range of just a short immediate displacement value. This in turn would give a programmer more freedom in that they would not have to worry about program length or certain parts of a program being out of reach of a branch. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to take Kawasaki's concept of selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register, and apply it to the system of Lee.

g) Lee has further taught that the processor is extended. More specifically, it is user-configured in that the user may configure whether to nullify or not nullify delay slot instructions and it is extended because the processor provides the capability of allowing for the nullification of the delay slot instruction, i.e., the nullification is interpreted as an extended feature. Lee has not explicitly taught that the processor is a RISC processor. However, it should be noted that Lee does not say that the processor is not a RISC processor. Official Notice is taken that RISC processors are well known and expected in the art. In addition, Heuring has shown advantages of a RISC design on pages 93-94. Some of these advantages include a reduced instruction set which is also less complex, fixed instruction length, completion of execution every clock cycle, simplified addressing modes, prefetching, and speculative execution. As a result, in order to realize some of these advantages, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Lee to be a RISC processor.

38. Referring to claim 63, Lee has taught a digital processor comprising:

Art Unit: 2183

a) a user-customized processor core having a multistage instruction pipeline, said core being adapted to decode and execute an instruction set comprising a plurality of instruction words.

Fig.5 shows a 4-stage pipeline that is further described in column 6, line 56, to column 7, line 39.

Furthermore, it is inherent that the processor will decode and execute multiple instructions (as established in the rejection of claim 1) from an instruction set. Finally, it should be realized that all processors are user-customized as processors are components designed by users.

b) a data interface between said processor core and an information storage device. It is inherent that in order for a processor to execute instructions, they must be stored in the processor's memory. Therefore, the instructions would be stored in an information storage device that is directly accessible by the processor.

c) an instruction set comprising a plurality of instruction words, at least one of said instruction words being a user-configurable jump instruction containing data defining a plurality of jump delay slot modes and at least one user-defined mode, said jump delay slot modes and at least one user-defined mode each being specified by the same portions of said data, said at least one user-defined mode not being predetermined in terms of function, said plurality of modes controlling the execution of instructions within said instruction pipeline of said processor core in response to said at least one jump instruction word within said instruction set. See Fig.2, component 102, and note the use of a branch (jump) instruction. Furthermore, Fig.3 shows that the branches are user-configurable in that a number of different modes which are achieved in part by the user. For example, the user can configure a jump instruction such that its delay slot instruction is always executed by turning the nullify bit off (see Fig.3 and Fig.1, field 507). In addition, the user can define the delay slot of an instruction to either execute or not execute by turning on the

Art Unit: 2183

nullify bit (see Fig.3 and Fig.1, field 507). With the nullify bit on, the functionality is not predetermined because whether or not the delay slot instruction executes or not is partially dependent on whether the associated branch is taken or not taken, and a branch outcome is determined as the program is running. So the functionality will be determined while the program is running. This is equivalent to at least one user-definable mode. It can be seen that this configurability and definability are specified by the same bits (i.e., the bits specifically used by branch instructions - Fig.2, fields 507 and 508, but specifically, the nullify bit).

d) wherein at least one of said plurality of instruction words comprises an op-code (column 3, lines 46-47), a plurality of fields (Fig.1), each of said fields comprising a plurality of bits (see column 3, lines 46-51), said at least one instruction word being encoded according to the method comprising:

d1) associating a first of said fields with a first data source. See column 3, lines 47-48 (component 503).

d2) associating a second of said fields with a second data source. See column 3, lines 48-49 (component 504).

d3) performing a logical operation using said first and second data sources as operands, said logical operation being specified by said op-code. See column 3, lines 51-55. In this case, the opcode specifies a compare and branch instruction where the comparison is performed between the contents of the two specified registers.

e) Lee has not taught that said at least one instruction word is used to generate a long immediate constant by selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said

Art Unit: 2183

portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register. However, Kawasaki has taught such a concept for branching purposes as well as for other instructions, such as move instructions. See column 51, lines 50-55. Kawasaki has disclosed a move instruction of the format: **mov #imm, Rn**, where the short immediate value specified by #imm is sign-extended to form a long immediate value which is then stored in the register specified by Rn. By sign-extending an immediate value, a portion (the sign bit) of the value is copied into the most significant bit positions of the long immediate value. For instance, if the #imm field specified a short 4-bit immediate value 1010, which is to be transformed into an 8-bit long immediate value, then 1010 would be sign-extended to 11111010, where the sign bit of the 4-bit value is copied into the 4 most significant bit positions of the long value. It also follows that the 4-bit value has been shifted. Note that initially, 1010 contained a 1 in the most significant bit position, a 0 in the second most significant bit position, a 1 in the third most significant bit position, and a 0 in the fourth most significant bit position. After sign-extending the 4-bit value, the same numbers become the fifth, sixth, seventh, and eighth most significant bits, respectively. Hence, they have been shifted. Furthermore, in column 42, lines 16-27, Kawasaki has disclosed that this type of move instruction is used to help branch to addresses out of the short immediate value's range. More specifically, if a branch needs to branch further than what the short displacement allows, then the destination address is moved to the register specified by the "mov" instruction and a "jmp" instruction (shown in column 48, lines 28-30) is used with to jump to the address stored in the register. A person of ordinary skill in the art would have recognized that this concept could be applicable in a system that is concerned with branching, such as Lee's. Such a concept would allow a branch instruction to branch to an

Art Unit: 2183

address outside of the range of just a short immediate displacement value. This in turn would give a programmer more freedom in that they would not have to worry about program length or certain parts of a program being out of reach of a branch. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to take Kawasaki's concept of selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register, and apply it to the system of Lee.

f) Lee has further taught that the processor is extended. More specifically, it is user-configured in that the user may configure whether to nullify or not nullify delay slot instructions and it is extended because the processor provides the capability of allowing for the nullification of the delay slot instruction, i.e., the nullification is interpreted as an extended feature.

39. Claims 70, 74, 78, and 82 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lee in view of Wirthlin, as applied above, and further in view of Aleksic et al., U.S. Patent No. 5,995,736 (herein referred to as Aleksic).

40. Referring to claim 70, Lee in view of Wirthlin has taught a core as described in claim 68. It has not been taught that said processor core is further user-customized by generating, through an automated process, test code associated with the customized description language model based on the at least one customized parameter. However, Aleksic has taught the concept of automatically generating test code to test VHDL code. See column 6, lines 57-62. Clearly, it is important to test VHDL code so that a model may be properly simulated and analyzed. Without

Art Unit: 2183

testing, it would be unclear as to whether a corresponding fabricated component will operate as intended. Automated testing allows for less time spent by the user in writing test code. In addition, it is obvious to automate as described in In re Venner, 120 USPQ 192 (CCPA 1958). As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Lee et al in view of Aleksic such that test code associated with the model is automatically generated.

41. Referring to claim 74, Lee in view of Wirthlin has taught a core as described in claim 72. Furthermore, claim 74 is rejected for the same reasons set forth in the rejection of claim 70.

42. Referring to claim 78, Lee in view of Wirthlin has taught a core as described in claim 76. Furthermore, claim 78 is rejected for the same reasons set forth in the rejection of claim 70.

43. Referring to claim 82, Lee in view of Wirthlin has taught a core as described in claim 80. Furthermore, claim 82 is rejected for the same reasons set forth in the rejection of claim 70.

Response to Arguments

44. Applicant's arguments filed on July 8, 2005, have been fully considered but they are not persuasive.

45. Applicant argues the novelty/rejection of claim 37 and 60-61 on pages 16 and 18 of the remarks, respectively, in substance that:

“As evidenced in the foregoing cite and the balance of Wirthlin, the processor of Wirthlin is implemented using a plurality of reconfigurable logic FPGAS. Wirthlin states in the above citation that as the number of these FPGAS in a given system increases, the ability to use conventional approaches such as schematic capture or hardware description languages (HDL) becomes unwieldy. Hence, Wirthlin purposely avoids using such conventional approaches, including HDL, since they require an innate knowledge by the user/designer.”

Art Unit: 2183

46. These arguments are not found persuasive for the following reasons:

a) The cited paragraph has been taken out of context in the paper. The paragraph relied upon to suggest that the reference teaches away from using HDL modeling is not stating that HDL modeling should not be used, it is stating that it becomes unwieldy when used to model a system with multiple FPGAs. Consequently, instead of implementing multiple FPGAs (which would make for an enormous HDL programming task), Wirthlin has developed a customizable FPGA in which a base instruction set is used (common instruction) and a user can define additional instructions to be executed (thereby not requiring additional FPGAs but simply customization of the existing FPGA). As a result, since multiple FPGAs are not used for every single application, the HDL modeling does not become unwieldy. It should be realized that regardless of the number of FPGAs, HDL and other description languages may still be used to model the system. It just becomes a larger task when the amount of FPGAs increases. So, Wirthlin is not teaching away from HDL modeling. He is simply stating that it becomes tougher to use when many FPGAs are present (hence the need for a customizable FPGA).

47. Applicant argues the novelty/rejection of claim 37 on page 16 of the remarks, in substance that:

“Wirthlin teaches an approach where a static and non-custom core is specified using FPGA assets, and then a “custom instruction set” is specified afterward and separately using the remainder of the FPGA assets. This is in stark contrast to applicant’s claimed invention, wherein the recited extension instruction is integrated into a common hardware description language (HDL) model of the processor.”

48. These arguments are not found persuasive for the following reasons:

Art Unit: 2183

a) The custom instruction set is still part of the overall design of the FPGA regardless of whether the core instruction set has already been determined. An FPGA design (including the extension set) may be modeled by HDL at any point in time. This allows for the simulation and testing of the core with the extension set and how they interact with one another.

49. Applicant argues the novelty/rejection of claims 37 and 59 on page 17 of the remarks, in substance that:

“Wirthlin (used as the basis for the Examiner's rejections of Claims 37-40) explicitly teaches away from a user-customized core; see page 26, first par. of Col. 1 wherein it is stated: “Hardware processors for a class of applications can be reused so users do not have to create a custom processor for each application,”.”

50. These arguments are not found persuasive for the following reasons:

a) See section 3.1.2 and note the custom instruction set. This set is customizable by the user and if the user customized the set, then the processor is still in a design phase, i.e., the instruction set is being designed by the user.

Conclusion

51. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire **THREE MONTHS** from the mailing date of this action. In the event a first reply is filed within **TWO MONTHS** of the mailing date of this final action and the advisory action is not mailed until after the end of the **THREE-MONTH** shortened statutory period, then the shortened statutory period

Art Unit: 2183

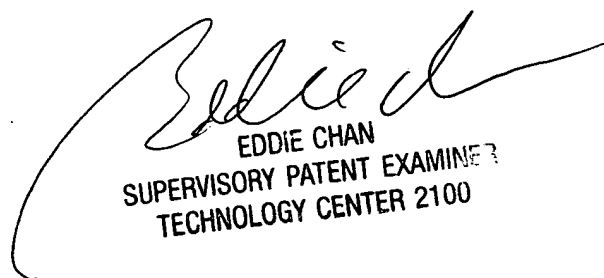
will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
September 7, 2005


EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100